Recitation 1: Spark (and Hadoop) Rutgers Fall 2021, Instructor: Maria Striki

Group #3

Aniqa Rahim - afr64@scarletmail.rutgers.edu
Shaan Kalola - spk103@scarletmail.rutgers.edu
Shivani Sunil - ss3013@scarletmail.rutgers.edu

HADOOP

TASK1: SPARK Code (3 points)

Go to $SPARK_HOME/core folder and navigate the core code of Spark (schedulers, etc). Do some searching to get to the core-site. If it does work, navigate in it and include a few print screens in your homework.
If you get obsessed about accessing … some core site then play around with this of Hadoop and send us a few printscreens: **See files**

Task 2: Hadoop-Wordcount Example in Java and Python: (8 points)

a) Compile and execute the wordcount code in the wordcount/java folder using the Makefile. Show the screenshot of the output.

b) Go to the wordcount/streaming directory. Write the wordcount map and reduce codes in python (may.py and reduce.py). Also edit the Makefile in the same directory to compile your code. Hand-in the code, makefile, and a screenshot of the output.

In the makefile that is included in the java folder, please make sure that these three lines which include -p have whitespace between -p and $(variable).

Solution: **See files**

Task 3: Write the corresponding Mappers and Reducers in Hadoop to find the 100 words that get most frequently used in a document: (5 + 5 + 4 = 14 points)

a) Write the corresponding Mappers and Reducers in Hadoop for Task 2. You may modify accordingly the ones you used for the WordCount and you may also modify the make file to include the text you have used.
b) Please submit the code, the solution, and your output. To test everyone' solution for correctness you may upload on hdfs and use the following document: http://www.gutenberg.org/files/1342/1342-0.txt

c) The results you have obtained so far are without the combiner. Think about how and where to add the combiner and re-compute your Mapper-Reducer jobs. Compare against the original version and have some print screens on the map and reduce input-output records, bytes, combine input-output records, shuffle input-output, and compare (Hadoop produces this information). **Hint:** under which circumstances (mathematical properties of the reduce function) can we use the reducer as a combiner?

In the makefile that is included in the java folder, please make sure that these three lines which include -p have whitespace between -p and $(variable).

Your Solution: **See files**

c) The combiner is placed after the mapper but before the reducer or before shuffle and sort. The reduce function must be associated and commutative in order for it to be used as a combiner. After adding the combiner the number of bytes read dramatically decreases as well as the number of bytes written, read operations, and write operations. In summation, the combiner dramatically decreases the time needed and resources used. Note the stdout file in task 3 part c folder contains print outs of all the data stored in the maps, combiners, and reducers after they perform their operations.